



كلية العلوم

القسم : الفيزياء

السنة : الثالثة

المادة : فيزياء حاسوبية

المحاضرة : الثالثة / نظري /

{{ مكتبة A to Z }}

مكتبة A to Z Facebook Group :

كلية العلوم

يمكنكم طلب المحاضرات برسالة نصية (SMS) أو عبر (What's app-Telegram) على الرقم 0931497960

2026

7

## المحاضرة الثالثة أوامر الرسم والبلاغات الشرطية والتكرارية

مدرّس المقرر:  
م. منار أحمد نجار

### الرسم البيانية

❖ يملك ماتلاب إمكانيات هائلة لعرض المتجهات والمصفوفات كرسوم، وتذييلها بالتعريفات كعنوان الرسم، وتسمية المحاور والمنحنيات، وبعد ذلك طباعة هذه الرسوم طباعة بجودة عالية.

#### أمر الرسم

❖ تقوم دالة **plot** بعدة مهام اعتمادا على مدخلاتها فلو كان **y** متجها فالأمر **plot(y)** يولد رسما للمتجه يفترض فيه أن **x** هي الرمز الدائلي

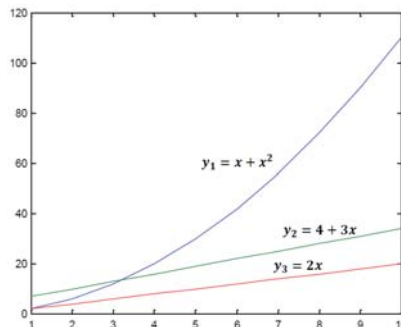
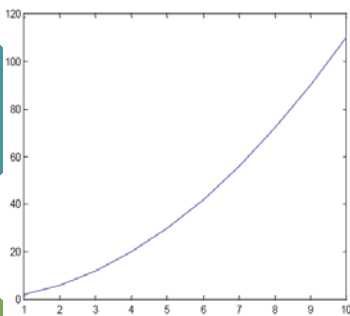
لكل عنصر في المتجه. أما الأمر **plot(x,y)** فيرسم العلاقة بين هذين المتغيرين في مستوى **(x,y)** حيث تمثل **y** قيم المحور العمودي و **x**

قيم المحور الأفقي فمثلا ، نستخدم الأوامر التالية لرسم هذه الدالة التربيعية:  $y = x + x^2$

```
x = 1:10;  
y = x + x.^2;  
plot(x,y)
```

❖ لاحظ أننا استخدمنا النقطة (.) قبل معامل الرفع للأس  $x^2$  ؛ لأننا نريد رفع كل

عنصر في المتجه **x** إلى الأس 2



❖ ويمكن رسم أكثر من علاقة في نفس الوقت بتكرار

المدخلات المطلوب رسمها كما هو موضح في المثال

التالي

```
y2 = 4 + 3*x;  
y3 = 2*x;  
plot(x,y,x,y2,x,y3)
```

## أمر الرسم

- ❖ وإذا كان المطلوب رسم أكثر من منحنى، فيقوم ماتلاب بشكل تلقائي بتلوين المنحنيات بألوان مختلفة. ومن الممكن أن يقوم المستخدم بتحديد لون مميز و (أو) علامات مميزة (+، ×) أو خط معين (متصل أو متقطع) لكل منحنى.
- ❖ والمثال التالي يوضح كيف يتم تلوين المنحنى الأول باللون الأصفر وخط منقط وعلامة +:

```
plot(x, y, 'y:+')
```

- ❖ لاحظ أن من الألوان المقبولة **c m y r g b w k** وتمثل الألوان الأسود والأبيض والأزرق والأخضر والأحمر والأصفر والأرجواني والأزرق الفاتح على التوالي. ومن الخطوط المقبولة - (متصل)، - - (مقطع)، : (منقط)، . (متصل منقط)، -- (مقطع منقط)، ومن العلامات المقبولة + ، o ، \* ، ×

>>> تنبيه <<<

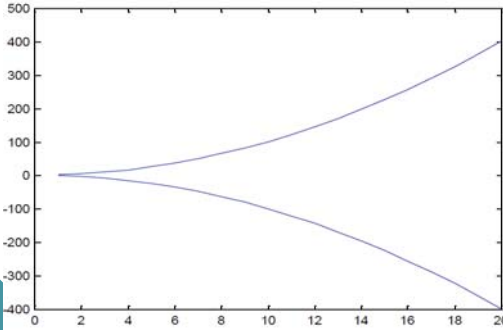
- ❖ بالإضافة إلى أمر الرسم **plot** يوجد أوامر أخرى لرسم أنواع مختلفة من الرسوم مثل الأوامر **bar** و **area** و **hist** و **pie** وغيرها.

## نافذة الرسم

- ❖ يقوم ماتلاب برسم الرسوم البيانية في نافذة الرسم. وتقوم دوال الرسم بفتح نافذة الرسم وإرسال الرسم إليها، أو إرسال الرسم إلى نافذة الرسم المفتوحة حالياً. وإذا كان هناك نافذة رسم موجودة، وأردت أن يكون الرسم الثاني في نافذة أخرى استخدم الأمر: **figure**
- ❖ فهذا الأمر يقوم بفتح نافذة رسم جديدة. أما الأمر: **figure(n)** فيقوم بإرسال الرسم إلى النافذة المفتوحة التي تحمل الرقم **n**

## إضافة رسم على الرسم الحالي

- ❖ يتيح الأمر **hold on** إضافة رسم منحنى أو منحنيات إلى نافذة رسم واحدة. فعندما تكتب: **hold on** يقوم ماتلاب بإضافة الرسم التالي لهذا الأمر فوق الرسم الموجود أصلاً في نافذة الرسم، مع تعديل المحاور إن تطلب الأمر ذلك (بدون هذا الأمر، يقوم ماتلاب بإزالة الرسم السابق ووضع الرسم الجديد مكانه). وهذا المثال يوضح عمل هذا الأمر:



```
x = 1:20;  
y1 = 2 - x.^2;  
plot(x, y1)  
hold on  
y2 = 2 + x.^2;  
plot(x, y2)
```

- ❖ تابع ماذا يحدث في نافذة الرسم حتى تحصل على هذا الرسم في الشكل الآتي

## تكوين رسوم صغيرة داخل نافذة الرسم

- ❖ تمكن دالة **subplot** من عرض أكثر من رسم في نافذة رسم واحدة، وكذلك طبعتها في صفحة

```
subplot(m,n,p)
```

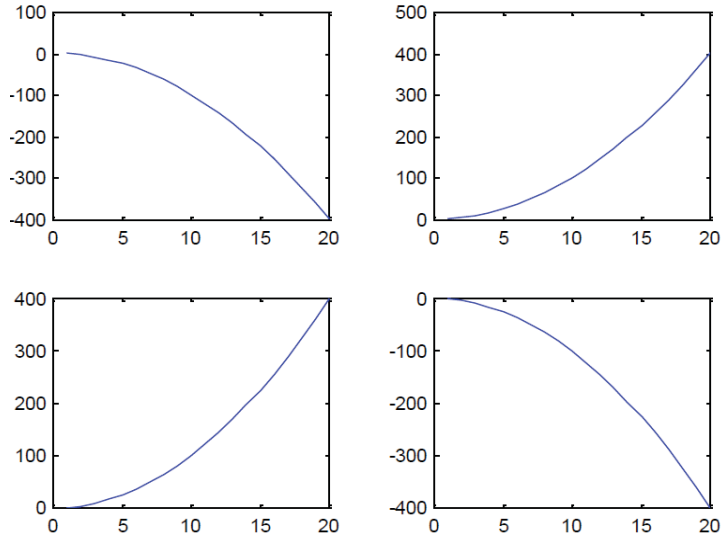
واحدة؛ فكتابة

- ❖ تقوم بتقسيم نافذة الرسم إلى مصفوفة ( **m x n** ) من الرسوم الصغيرة، مع إعطاء المكان **p** للرسم التالي؛ حيث يتم ترتيب الرسوم في النافذة من اليسار إلى اليمين، ثم من الأعلى للأسفل؛

## تكوين رسوم صغيرة داخل نافذة الرسم

❖ فعلى سبيل المثال، تقوم الأوامر التالية برسم أربعة رسوم صغيرة في نافذة واحدة، موضحة في الشكل

```
x = 1:20;  
y1 = 2 - x.^2;  
y2 = 2 + x.^2;  
y3 = x.^2;  
y4 = -x.^2;  
subplot(2,2,1); plot(x,y1);  
subplot(2,2,2); plot(x,y2);  
subplot(2,2,3); plot(x,y3);  
subplot(2,2,4); plot(x,y4);
```



<<< تنبيه >>>

يمكن كتابة أكثر من أمر في سطر واحد بشرط أن يفصل بينهما بفاصلة عادية ( , ) أو منقوطة ( ; )

## التحكم بالمحاور

❖ للدالة **axis** العديد من الخيارات التي تسمح بالتحكم في مقياس واتجاه المحاور. وبدون تحديد أي خيارات يقوم ماتلاب باختيار قيمة المحاور مستخدماً القيم الكبرى والصغرى في البيانات المراد رسمها.

❖ لكن تحديد هذه الخيارات من قبل المستخدم يجبر ماتلاب على اعتمادها في الرسم. فالأمر: **axis([xmin xmax ymin ymax])**

❖ يلزم ماتلاب باستخدام القيم بين القوسين لبداية ونهاية محوري **x** و **y** أما الأمر: **axis square**

❖ فيجعل الفرق بين كل قيمتين في المحورين متساوية.

❖ والأمران التاليان يتحكمان في مسألة ظهور أو اختفاء المحاور من الرسم:

```
axis on  
axis off
```

❖ أما الأمران:

```
grid off  
grid on
```

❖ فيتحكمان في ظهور الخطوط الأفقية والرأسية في الرسم.

## تسمية عنوان الرسم والمحاور

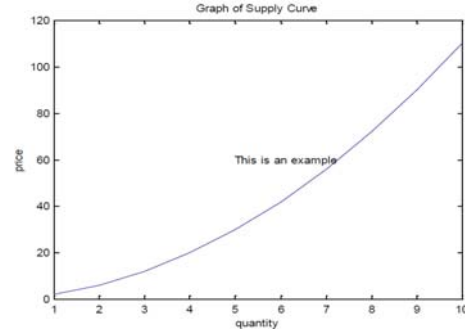
❖ تضيف الدوال الثلاث (**xlabel,ylabel,zlabel**) أسماء المحاور للرسم.

❖ أما الدالة **title** فتضيف عنوان الرسم في أعلاه.

❖ وتسمح الدالة **text** بكتابة نص في مكان ما في نافذة الرسم؛ والمثال التالي يوضح عمل هذه الدوال:

## تسمية عنوان الرسم والمحاور

```
x = 1:10;
y = x + x.^2;
plot(x,y)
xlabel('quantity')
ylabel('price')
title('Graph of Supply Curve')
text(5,60,'This is an example')
```



❖ لاحظ أن النصوص توضع دائما بين علامتي تنصيص، ولاحظ أيضا أننا طلبنا من

ماتلاب عند استخدام الأمر **text** أن يضع النص عند الإحداثي ( 5,60 )

### مفتاح الرسم البياني

❖ يضع الأمر **legend** صندوق صغير داخل الرسم، يبين فيه

مسمى كل منحنى يظهر في نافذة الرسم. **legend('Supply');**

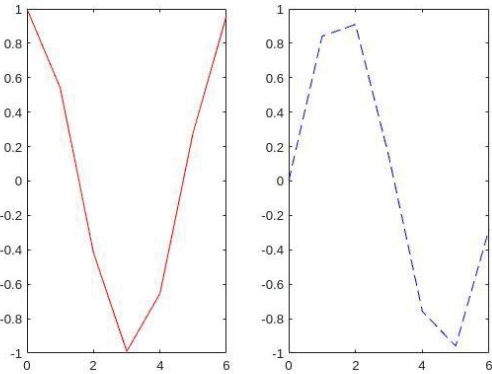
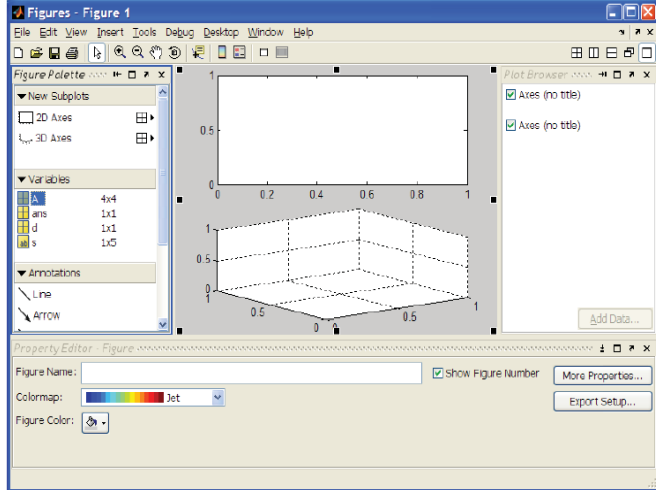
### أدوات الرسم

❖ تتضمن بيئة ماتلاب أدوات رسم إضافية تسهل بشكل كبير تحرير

رسومات ماتلاب حسب تفضيلات المستخدم، ويمكن الوصول لهذه

الأدوات من خلال نافذة الرسم؛ أو من خلال الأمر، **plottools**.

ويبين الشكل نافذة أدوات تحرير الرسومات



### تمرين 1:

اكتب بلغة ماتلاب التعليمات اللازمة لرسم التتابع  $y_1 = \cos(t), y_2 = \sin(t), y_3 = x^2, y_4 = e^x$  ضمن نافذة واحدة مقسمة لأربعة رسوم صغيرة، حيث تأخذ  $t$  قيمها ضمن المجال من 0 إلى  $2\pi$ ، وتأخذ  $x$  قيمها ضمن المجال من 0 إلى 20، مع إعطاء كل تابع لون معين وخط معين من اختيارك بحيث يكون لكل تابع خط ولون مختلف عن الآخر.

```
t = 0:2*pi;
x = 0:20;
```

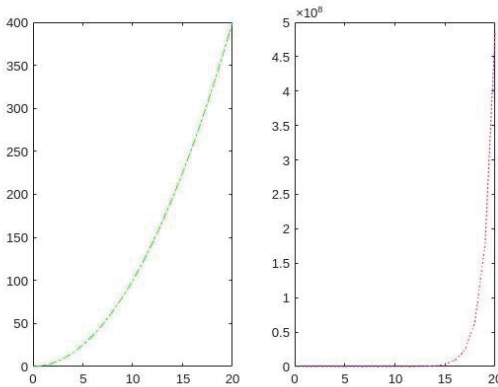
```
y1 = cos(t);
y2 = sin(t);
y3 = x.^2;
y4 = exp(x);
```

```
subplot(2,2,1);
plot(t, y1, 'r-');
```

```
subplot(2,2,2);
plot(t, y2, 'b--');
```

```
subplot(2,2,3);
plot(x, y3, 'g-.');
```

```
subplot(2,2,4);
plot(x, y4, 'm:');
```



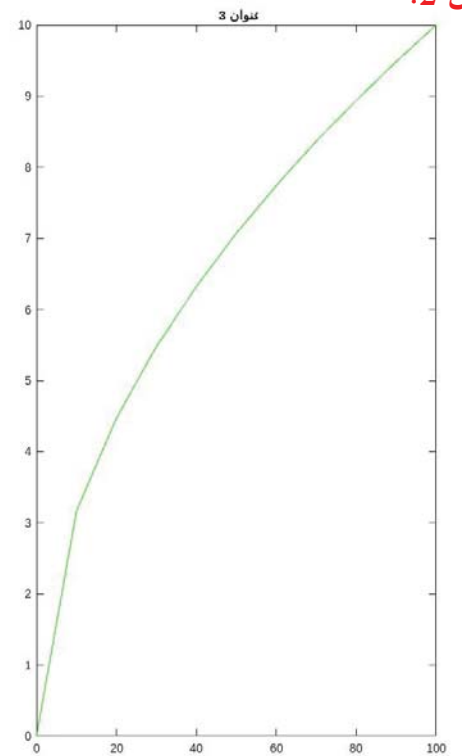
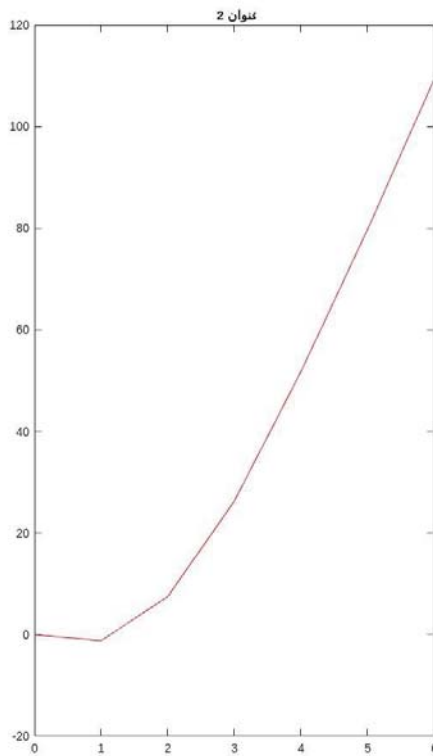
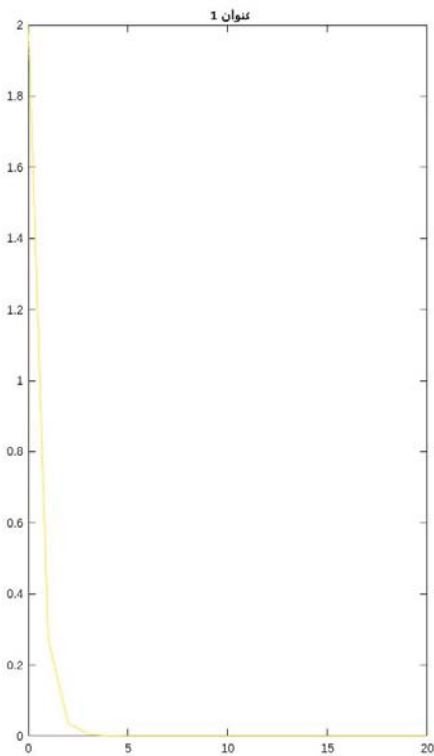
## تمرين 2:

اكتب بلغة ماتلاب التعليمات اللازمة لرسم كل من التوابع التالية بحيث يظهر كل تابع في نافذة منفردة، وله لون فريد، مع عنوان فريد لكل رسم:

$y_1 = 2e^{-2x}$ ,  $y_2 = 3t^2 - 5 \sin t$ ,  $y_3 = \sqrt{|-z|}$   
حيث تأخذ  $t$  قيمها ضمن المجال من 0 إلى  $2\pi$ ، وتأخذ  $x$  قيمها ضمن المجال من 0 إلى 20، وتأخذ  $z$  قيمها ضمن المجال من 0 إلى 100 بخطوة 10.

```
x=0:20;  
y1=2*exp(-2*x);  
figure  
plot(x,y1,'y')  
title('عنوان 1')  
  
t=0:2*pi;  
y2=3*t.^2-5*sin(t);  
figure  
plot(t,y2,'r')  
title('عنوان 2')  
  
z=0:10:100;  
y3=sqrt(abs(-z));  
figure  
plot(z,y3,'g')  
title('عنوان 3')
```

## تمرين 2:



## التحكم بالانسياب والتكرار

- ❖ يمكن التحكم بالانسياب بواسطة البلاغات الشرطية، والتحكم في تكرار تنفيذ أمر أو مجموعة من الأوامر بواسطة الحلقات التكرارية.
- ❖ يحتوي ماتلاب عدة طرق للتحكم في البلاغات الشرطية مع أو بدون الحلقات التكرارية، أهمها:

- 1 . بلاغ الشرط.
- 2 . بلاغ الانتقال.
- 3 . التكرار المحدود.
- 4 . التكرار غير المحدود.
- 5 . بلاغ وقف التكرار.

### بلاغ الشرط

- ❖ تقوم الجملة الشرطية (**if**) بتقييم التعبيرات المنطقية؛ ومن ثم تنفيذ أمر أو مجموعة من الأوامر إذا كانت التعبيرات المنطقية صحيحة.
- ❖ وتستخدم كلمة **end** لتحديد نهاية عمل الجملة الشرطية؛ ومن أبسط صيغها:

```
A = 2;  
if A == 2  
disp('A تساوي 2')  
end
```

- ❖ في هذا المثال يتحقق ماتلاب من صحة التعبير المنطقي، فإن كان صحيحاً (وهو صحيح في هذا المثال) يتم عرض الجملة (A تساوي 2) على الشاشة باستخدام الأمر **disp**. ولو لم تكن A=2، فلن يعرض شيئاً.
- ❖ ويبين هذا المثال أن ماتلاب يمكن أن يتعامل مع نصوص اللغة العربية.

### بلاغ الشرط

- ❖ ويمكن استخدام الشرطين **elseif** و **else** لتنفيذ أوامر أخرى في حالة عدم صحة شرط التعبير المنطقي الخاص بالشرط **if** مثال:

```
A = 3;  
if A == 2  
disp('A تساوي 2')  
elseif A == 0  
disp('A تساوي صفر')  
else  
disp('A لا تساوي 2')  
end
```

- ❖ في هذا المثال يتحقق ماتلاب من صحة تعبيرين منطقيين، فإن كان أحدهما صحيحاً عرض الجملة المناسبة على الشاشة.
- ❖ أما إذا كانا غير صحيحين فينفذ الأمر الذي يلي **else**، وهو عرض رسالة (A لا تساوي 2).
- ❖ ومن المهم أن تدرك أن هذه الشروط ومعاملات العلاقات المنطقية تعمل مع الأعداد المفردة بطريقة مختلفة عن المصفوفات.
- ❖ فللتأكد من تحقق المساواة بين متغيرين مفردين يمكن أن تستخدم: **if a == b, ...**

- ❖ أما في حالة المصفوفات فالشرط يتأكد من تساوي كل عنصر مع ما يقابله، وليس تساوي المصفوفات ككل (بشرط تساوي حجم المصفوفات). والطريقة السليمة للتحقق من تساوي المصفوفات هي استخدام دالة **isequal** على هذا النحو: **if isequal(A,B), ...**

### بلاغ الانتقال بين الحالات

- ❖ تنفذ العبارة **switch** مجموعة من العبارات اعتماداً على قيمة المتغير أو التعبير.
- ❖ دور الكلمتين **case** و **otherwise** هو تقسيم هذه المجموعات إلى حالات مختلفة.
- ❖ أول حالة تتطابق مع الشرط يتم تنفيذها فقط. ويجب أن يكون هناك كلمة **end** كالمعتاد لتحديد متى يتوقف ماتلاب عن التحقق من الشرط.

## بلاغ الانتقال بين الحالات

```
A = 3;
switch A
    case 2
disp('A تساوي 2')
    case 0
disp('A تساوي صفر')
    otherwise
disp('A لا تساوي 2')
end
```

❖ ويمكن التعبير عن مثال بلاغ الشرط السابق باستخدام هذا الشرط:

## التكرار لعدد محدود من الدورات

❖ تقوم **for** بتكرار تنفيذ مجموعة من التعبيرات لعدد محدد من المرات. ويجب أن يقابلها ما يفيد انتهاء التعبيرات الداخلة في حلقة التكرار وهو كالمعتاد كلمة **end** مثال:

```
n = 3;
for i = 1:10
    r(i) = i*n;
end
```

```
r =
     3     6     9    12    15    18    21    24    27
    30
```

هذه الحلقة التكرارية تقوم بتكوين متجه صف مكون من عشرة عناصر .

وفي كل تكرار تقوم الحلقة بتكوين عنصر جديد في هذا الصف يساوي الرمز الدليلي **i** مضروباً في العدد **n** الذي يساوي 3. ويتم حفظ النتيجة في المتجه **r**

## التكرار لعدد محدود من الدورات

```
m = 5; n = 3;
for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
H =
```

```
0.5000    0.3333    0.2500
0.3333    0.2500    0.2000
0.2500    0.2000    0.1667
0.2000    0.1667    0.1429
0.1667    0.1429    0.1250
```

❖ تعد طريقة كتابة التكرار مع ترك مسافات مناسبة عند البرمجة لتسهيل القراءة وللتمييز بين مختلف الحلقات التكرارية. يقوم محرر ماتلاب بهذه المهمة بشكل تلقائي.

❖ توضح الحلقات التكرارية المتداخلة في المثال التالي فائدة ترك المسافات:

فتقوم هاتان الحلقتان التكراريتان المتداخلتان بتكوين المصفوفة **H** ذات الأبعاد  $(5 \times 3)$ ، وكل عنصر فيها يساوي الواحد مقسوماً على جمع الرمزین الدليليين الخاص بذلك العنصر.

## التكرار لعدد غير محدود من الدورات

❖ تقوم **while** بتكرار تنفيذ تعبير أو مجموعة من التعبيرات عدداً لا نهائياً من المرات حتى يتحقق الشرط المنطقي المطلوب تحققه.

وكالعادة يجب أن يوجد كلمة **end** لتحديد الجزء من البرنامج الداخل في عملية التكرار؛ والمثال التالي يوضح عمل هذه الحلقة التكرارية:

```
a = 0.1;
while a^2 < 100
    a = a + 1;
end
a
```

فالمطلوب هنا أن يفترض ماتلاب ابتداءً أن قيمة المتغير **a** هي 0.1 ،

ثم يبدأ بإضافة العدد 1 إليه ويجمع الناتج مع 1 مرة أخرى،

ويستمر في هذه العملية مادام أن مضروب **a** في نفسها أقل من 100 . وستلاحظ أن التكرار سوف

يتوقف عندما تصل قيمة **a** إلى 10.1 حيث إن مضروب هذا العدد في نفسه يتجاوز 100

```
a =
    10.1000
```

## بلاغ وقف التكرار

- ❖ تستخدم كلمة **break** مع **for** و **while** لوقف عملية التكرار،
- ❖ وتعد عملية وقف التكرار مهمة للخروج من التكرار عندما لا يكون هناك فائدة من استمراره (خاصة في حالة التكرار غير المحدود)؛
- ❖ والمثال التوضيحي يبين ذلك:

```
a = 0; b = 3;
while b-a > 0.01*b
    x = (a+b)/2;
    if x == 1.5,
        break
    end
end
```

- ❖ فنحن نعلم من قيم **a** و **b** أن قيمة المتغير **x** تساوي **1.5** وأن  $0.01*b$  لا يمكن أن تزيد عن **b-a**، ولهذا سيستمر التكرار إلى ما لا نهاية إذا لم نقم بتوقيفه باستخدام بلاغ وقف التكرار.

### تمرين:

أوجد خرج البرنامج التالي:

```
x= 5; y= 10; z=0;
while y>=x
    z=z+y;
    y=y-1;
end
z
```

#### التكرار الأول:

```
while 10>=5
z=0+10=10
y=10-1=9
```

#### التكرار الثاني:

```
while 9>=5
z=10+9=19
y=9-1=8
```

#### التكرار الثالث:

```
while 8>=5
z=19+8=27
y=8-1=7
```

#### التكرار الرابع:

```
while 7>=5
z=27+7=34
y=7-1=6
```

#### التكرار الخامس:

```
while 6>=5
z=34+6=40
y=6-1=5
```

#### التكرار السادس:

```
while 5>=5
z=40+5=45
y=5-1=4
```

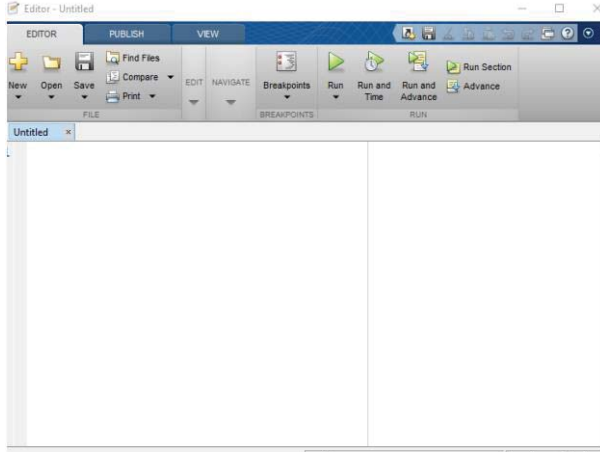
#### التكرار السابع:

```
while 4>=5
    غير محقق
نخرج من حلقة
و يكون الخرج
Z=45
```

## M-file

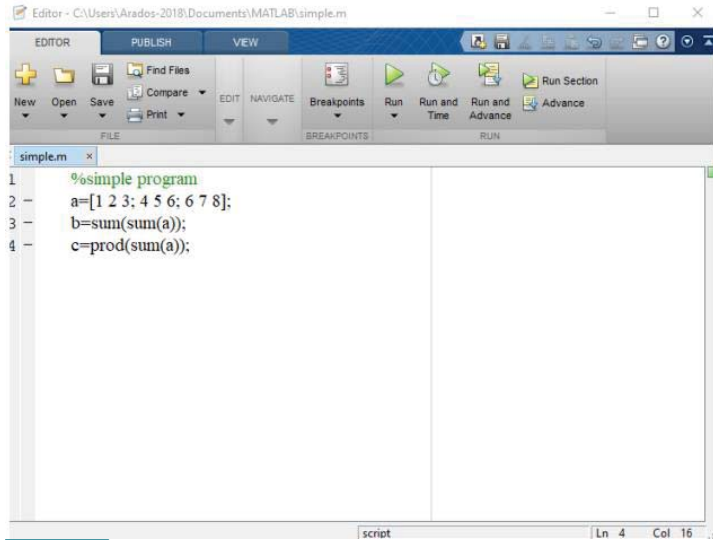
هي وسيلة لإدخال الأوامر ولكن ليس من خلال نافذة الأوامر، ولكن ماذا قد يختلف في هذه الوسيلة الجديدة في إدخال الأوامر؟

- 1 - في عملية إدخال الأوامر التي كنا نستخدمها، إذا أردنا تعديل عنصر أو أكثر كان يجب إعادة إدخال الأمر من جديد.
- 2- إذا وجد خطأ، فيجب كتابة الأمر من جديد .
- 3 - إذا كتبنا برنامج كبير، وأردنا إعادة العملية مرة أخرى يجب إدخال جميع الأوامر من جديد وبنفس الترتيب.
- 4 - إذا حدث خطأ في ترتيب الأوامر لهذا البرنامج الكبير ستقوم بإعادة إدخال الأوامر من البداية مرة أخرى.
- 5 - يصعب عمل عملية تصحيح للأخطاء .



وهذا بالطبع يستغرق وقتاً كبيراً بالإضافة إلى الملل الذي يحدث للمستخدم ولحل هذه المشكلة تم عمل ما يسمى بالملفات الميمية أو M-File والتي تعطي القدرة على كتابة البرنامج كاملاً وبدون تشغيل، وبعد الانتهاء منه يتم تشغيله، هذه الخاصية تعطي القدرة على تعديل القيم دون الحاجة إلى كتابتها مرة أخرى، أو إعادة إدخال الأوامر التي تعتمد على هذا الأمر. فكيف يتم تشغيل تلك الخاصية؟ يمكن تشغيل هذه الخاصية من خلال `new script`.

## M-file



ولكن عند الضغط على زر التشغيل، سيطلبك الماتلاب بحفظ البرنامج، ويشترط الآتي عند حفظ البرنامج:

- 1 - أن لا يبدأ بأرقام
  - 2 - أن لا يكون أمراً معروفاً
  - 3 - أن لا يحتوي الاسم على مسافات فاصلة
  - 4 - أن لا تحتوي على رموز خاصة مثل \*, &, -, +
- يجب مراعاة تلك الشروط وإلا لن يقوم الماتلاب بتنفيذ البرنامج. منقوم بتنفيذ المثال المكتوب:

بعد عملية الحفظ والتنفيذ يتم ظهور البارامترات في نافذة ال workspace كما يتم ظهور تنفيذ الأوامر ضمن نافذة الأوامر في حال عدم وضع فاصلة منقوطة. وهكذا يمكن في كل مرة إعادة تنفيذ البرنامج مع قيم جديدة.

في كل عملية تحديث للبرنامج ستظل قيم البرنامج القديم موجودة، ولحل المشكلة يتم وضع الأمر `clc` في أول كل برنامج.

كما يجب وضع الأمر `Clear` بعد الأمر `clc` بحيث يقوم بمسح أي قيمة سابقة في جميع البرامج من أي برنامج آخر في `Workspace`.

## الملفات الميمية: ملفات الدوال

```
function y = mean(x,dim)
%MEAN Average or mean value.
% For vectors, MEAN(X) is the mean value of the elements in X. For
% matrices, MEAN(X) is a row vector containing the mean value of
% each column. For N-D arrays, MEAN(X) is the mean value of the
% elements along the first non-singleton dimension of X.
%
% MEAN(X,DIM) takes the mean along the dimension DIM of X.
%
% Example: If X = [0 1 2
%                 3 4 5]
%
% then mean(X,1) is [1.5 2.5 3.5] and mean(X,2) is [1
%                                                    4]
%
% See also MEDIAN, STD, MIN, MAX, COV.
%
% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.13 $ $Date: 1997/11/21 23:23:55 $
```

- ❖ الملفات الدالية هي ملفات ميمية يمكن أن تقبل مدخلات وتولد مخرجات، ويفضل أن تكتب الدوال التي تحتاج إليها بشكل متكرر في ملف ميمي دالي.
- ❖ يجب أن يكون اسم الدالة داخل الملف متطابقاً مع اسم الملف، والاختلاف الرئيس كما أشرنا بين هذا النوع وملفات الأوامر المتتابعة هو في كيفية تعامل ماتلاب مع متغيراتها؛
- ❖ فالملفات الدالية تتعامل مع متغيرات داخلية خاصة بها، ومنفصلة عن المتغيرات الموجودة في ساحة العمل.
- ❖ ومثال جيد لهذا النوع من الملفات الميمية ملف **mean.m** والذي يحتوي على دالة تقوم بحساب المتوسط الحسابي

```
if nargin==1,
% Determine which dimension SUM will use
dim = min(find(size(x)~=1));
if isempty(dim), dim = 1; end

y = sum(x)/size(x,dim);
else
y = sum(x,dim)/size(x,dim);
end
```

لاحظ أنه يمكنك الاطلاع على محتويات أي ملف ميمي بكتابة **type** تتبعه باسم الملف. فكتابة:

**type mean**

يعرض على الشاشة:

## الملفات الميمية: ملفات الدوال

❖ ويظهر من هذا الملف الأقسام الثلاثة للملف الدالي:

1 . **سطر تعريف الدالة.** وهو يبدأ بالكلمة الاصطلاحية **function** يتبعها الطريقة التي سوف تستخدم بها الدالة . **y=mean(x,dim)**

2 . **مقطع نصوص المساعدة،** ونلاحظ أن هذه النصوص مسبوقة بالعلامة % للدلالة على أن هذه النصوص تعد تعليقات لا تنفذ وفي هذه

التعليقات يتم وصف الدالة وشرح طريقة استخدامها، وهذه التعليقات هي أيضا التي تظهر على الشاشة عند طلب المساعدة بكتابة:

**help mean**

3. **متن الدالة.** وهي بقية الأسطر التي تلي التعليقات في الملف. وهي أسطر قابلة للتنفيذ لأنها جزء من لغة ماتلاب؛ وفي هذه الأسطر يتم

```
A = [2 4;6 8]
A =
     2     4
     6     8
mean(A)
ans =
     4     6
mc = mean(A)
mc =
     4     6
mr = mean(A,2)
mr =
     3
     7
```

استخدام المدخلات في عمليات حسابية معينة للحصول على مخرج الدالة.

❖ لاحظ أن المتغيرات **dim,y,x** تعد متغيرات داخلية خاصة بالملف الدالي (وإن كان يوجد متغيرات مماثلة لها في الاسم في ساحة العمل) فلا يشترط أن تكون موجودة أصلاً في ساحة العمل، كما أنها لا تبقى فيه بعد استخدام هذه الدالة.

❖ يوضح هذا المثال أحد مزايا الملفات الميمية الدالية وهي قابلية عدد المدخلات للتغير؛ فيمكن أن يكون هناك مدخل واحد أو اثنان.

❖ ففي الحالة الأولى لم يتم وضع متغير لحفظ النتيجة؛ ولذا تحفظ النتيجة في المتغير **ans** كالمعتاد.

❖ أما في الحالة الثانية فهناك مدخل واحد للدالة، وفي هذه الحالة تقوم الدالة بوضع قيمة معطاة للمدخل الآخر، والقيمة

المعطاة للمدخل الثاني في هذه الدالة هي **1** (ويعني احسب متوسط الأعمدة).

❖ وفي الحالة الثالثة حددنا المدخل الثاني بالعدد **2** لكي نطلب من الدالة حساب متوسط الصفوف بدلاً من الأعمدة.

## type mean

يُستخدم لعرض كود المصدر (Source code) الخاص بالدالة `mean` في نافذة الكوماند ويندوز.

### ماذا يعرض تحديدًا؟

- لو كانت الدالة `mean` هي دالة m-file (مكتوبة بكود MATLAB)، فسيعرض محتويات ملف `mean.m` بالكامل في نافذة الأوامر.
- أما لو كانت `mean` دالة مدمجة (built-in) — وهي كذلك في الإصدارات الحديثة — سيعرض رسالة مثل:

```
matlab
Edit ✎ Copy 📄
'mean' is a built-in function.
```

بمعنى أن الكود الخاص بها مكتوب بلغة C أو ++C داخل النظام ولا يمكن للمستخدم الإطلاع عليه من MATLAB مباشرة.

## برمجة ملفات الدوال الميمية

- ❖ يستطيع المستخدم برمجة أي دالة يرغبها باستخدام مفهوم ملفات الدوال الميمية أعلاه؛
- ❖ على سبيل المثال: إذا أراد المستخدم برمجة دالة تحسب قيمة الدالة  $\sqrt{x^2 + z^2}$  عند أي قيمتين للمتغيرين `x` و `z`، فيمكنه بكل سهولة كتابة ما يلي في ملف نصي:

```
function y = myfile(x,z)
y=sqrt((x.^2)+(z.^2));
```

- ❖ ثم حفظه في ملف نصي باسم `myfile.m` في المسار الحالي لماتلاب، ثم تنفيذه بكتابة الأمر:

```
x=7;
z=3;
y=myfile(x,z)
```

y =

7.6158

- ❖ لتحصل على نتيجة تقييم الدالة عند قيمتي `x=7` و `z=3`

# Thank you



Eng. Manar Najjar  
[manarnajjar@tartous-univ.edu.sy](mailto:manarnajjar@tartous-univ.edu.sy)



مكتبة  
A to Z